docudata

Direct access • Total control

EDC Data Capture User Manual

© 2010 DocuData Software Corporation

EDC Data Capture User Manual

© 2010 DocuData Software Corporation

All rights reserved. No parts of this work may be reproduced in any form or by any means - graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems - without the written permission of the publisher.

Products that are referred to in this document may be either trademarks and/or registered trademarks of the respective owners. The publisher and the author make no claim to these trademarks.

While every precaution has been taken in the preparation of this document, the publisher and the author assume no responsibility for errors or omissions, or for damages resulting from the use of information contained in this document or from the use of programs and source code that may accompany it. In no event shall the publisher and the author be liable for any loss of profit or any other commercial damage caused or alleged to have been caused directly or indirectly by this document.

Printed: September 2010 in Montreal, QC, Canada

Publisher

DocuData Software Corporation

Technical Editor

Jeffrey Ritchie

Cover Designer

Erick Paquin

Table of Contents

Part I Introduction 4

1 System Requirements & Licensing ... 5

Part II Creating shortcuts for creating job definitions or performing data capture 6

Part III Defining Data Capture Jobs 8

Part IV Defining Data Fields to Capture 12

Part V Distributing and Configuring EDC Data Capture for End Users 14

Part VI Performing Data Capture 15

Part VII Retrieving Captured Data 17

Part VIII Using Scripts to Enhance Your Data Capture 18

1 EDC Data Capture Scripting Fundamentals ... 18

Examples of scripts for getting look-up values ... 18

Examples of scripts for validating values ... 19

Examples of scripts for transforming values ... 20

2 EDC Data Capture API Reference ... 21

Part IX Appendix 39

1 Mask values ... 39

2 Editing the interface wording ... 39

© 2010 DocuData Software Corporation

1 Introduction

The **EDC Data Capture** software is a self-contained (no installation required) program that streamlines and validates systematic data entry.

Output:
EDC Data Capture produces output in a simple and widely-used text format known as CSV, or "comma separated value". CSV is the preferred format for importing data into many applications, including **DocuData's EDC RC** and **ActiveFile Software**. CSV can be opened and edited with any text editor or spreadsheet application.

Basic Steps:
1. **Define Jobs:** The first step in working with EDC Data Capture is to create "**job definitions**", or define a job. A job only needs to be defined once. A job definition determines:
- Where the data will be saved
- What values will be captures
- What validation and transformations are applied

2. **Capture Data:** Once a job has been defined, it can be run repeatedly in **"capture mode"**. Users performing data capture do not require knowledge of how to define jobs. Simply distribute the self-contained program with the job definition to your users, set them up with the appropriate shortcuts, and they are ready to start.

3. **Retrieve and use the data:** The captured data is automatically stored in the location specified by the job definition, with a file name that is automatically calculated from the job name, the user name, and the date and time of data capture.

Validation and transformation scripting:
The validations and transformations are done with scripts (Object Pascal language). These scripts are not mandatory for simple data capture but can greatly enhance the possibilities of this software if used. DocuData's custom scripting services can be a very time and cost-effective solution. You may also consult the chapter in this manual on Using Scripts to Enhance Your Data Capture 18.

This manual will guide you through setting up and working with EDC Data Capture.

If you have any comments or suggestions to help us improve this software and/or manual please send an email to support@docudatasoft.com.

© 2010 DocuData Software Corporation

1.1 System Requirements & Licensing

System Requirements

Basic requirements for working with EDC Data Capture are minimal. Virtually any recent Windows-based computer can run it. It remains to be tested on other operating systems.
An internet connection is needed every 90 days to validate the software's license with DocuData's servers. (see licensing requirements below).
Custom scripts 18 for user-defined data capture jobs may also require specific network connections.

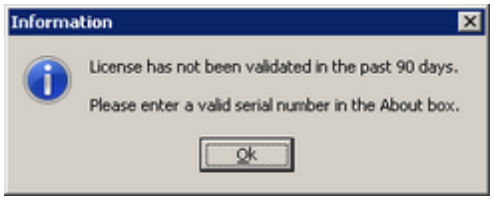
Licensing

The **EDC Data Capture** software does not require a license for use in creating create job definitions, but it does require a valid license to capture data. Contact support@docudatasoft.com if you require a license.

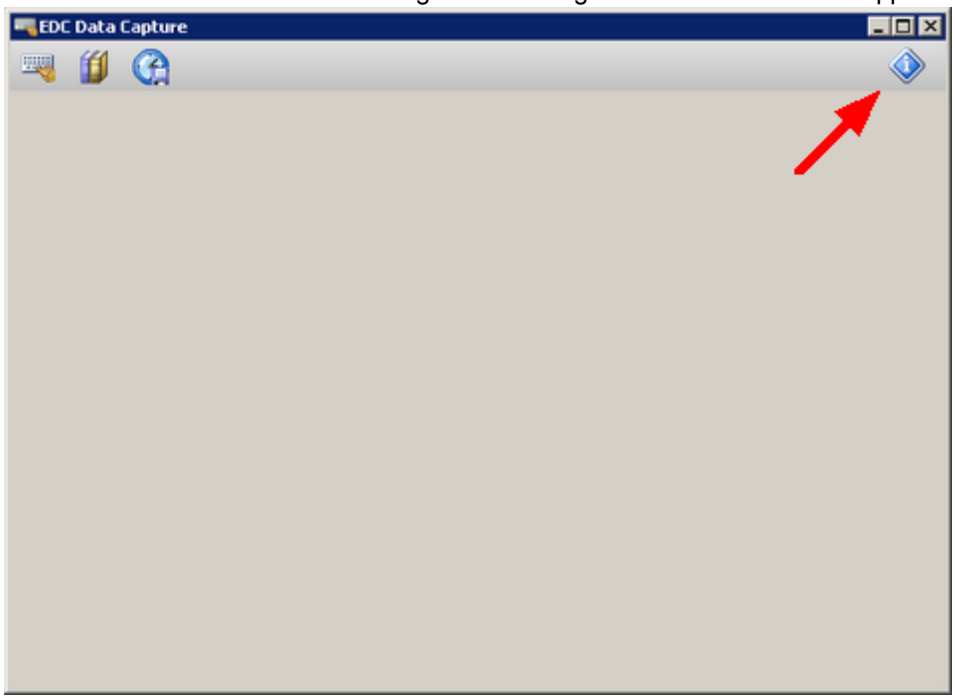
EDC Data Capture attempts to validate the license with DocuData's servers via the internet. It will remain active for up to 90 days from the last successful attempt, enabling you to use it without an internet connection.

How to enter a valid license into the EDC Data Capture software

When you first open the software (by running the "EdcDataCapture.exe" file), if the serial number has not been validated in the past 90 days you will see the following message:

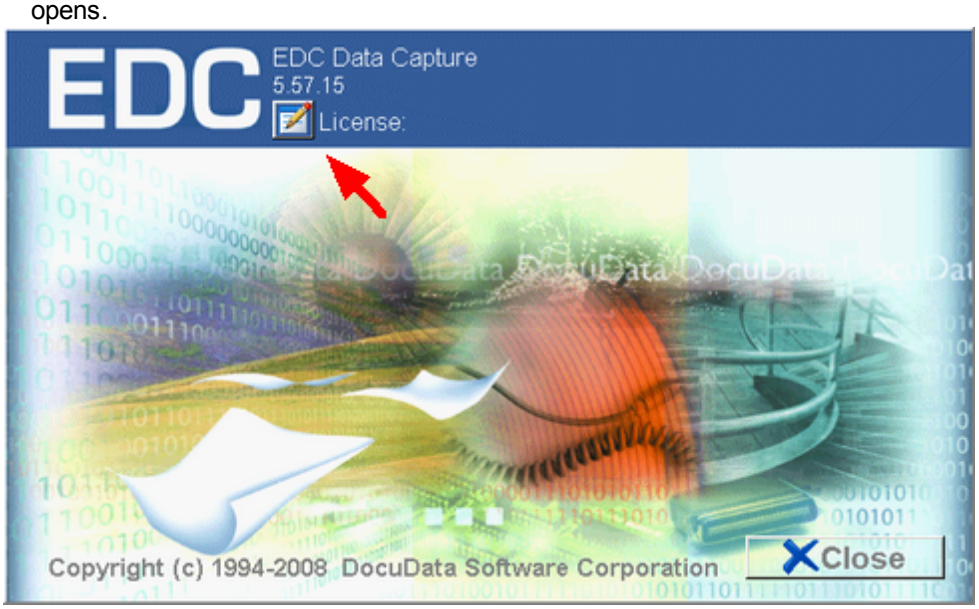


1. Click the Ok button to acknowledge the message. The main screen will appear.

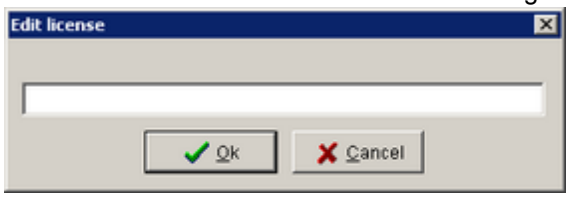


© 2010 DocuData Software Corporation

2. Click the information button in the top right corner. A screen with information about the application opens.



3. Click the button to edit the license. A dialogue box appears/



4. Paste in your new license, then click the **Ok** button to accept. The system will verify the license with DocuData. EDC Data Capture is now licensed for use.

2 Creating shortcuts for creating job definitions or performing data capture

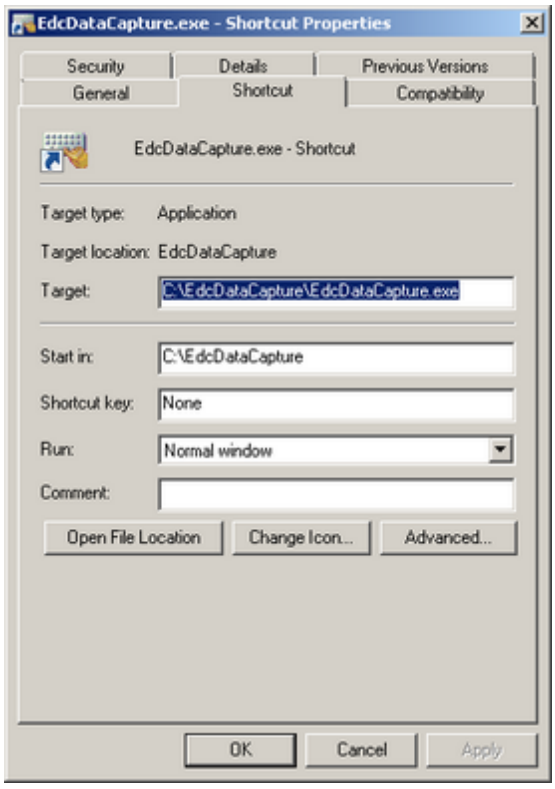
When you are defining new data capture jobs, you will take advantage of many capabilities. However, when your users are performing data capture, you will typically want the interface to be as simple as possible. For this reason, EDC enables you to configure the interface using command-line parameters as defined below.

Command-Line Parameters

Command line parameters are bits of text that are added to a call to a program and affect its behavior. They may be optional or required. In practice, you can add command line parameters by editing a short-cut so that it will call the program this way each time.

1. Create a shortcut to EdcDataCapture.exe, for example by browsing to find this file, then **right-clicking** on it and selecting "**Create shortcut**" from the menu.
2. Edit the shortcut. In windows you can do this by right-clicking on the shortcut, and selecting " **properties**" from the menu. the following window appears.

© 2010 DocuData Software Corporation



3. To the right of "EdcDataCapture.exe", you can type or paste in any of the command-line parameters that apply (see below for available options). Multiple parameters can be added sequentially, separated by spaces. eg:*C:\EdcDataCapture\EdcDataCapture.exe username="John Doe" capturefile="c:\definition_jobs\mydef.def*".
4. Click **OK** to save changes. You may rename the shortcut if desired. Clicking on this shortcut will now cause EdcDataCapture to be run with the desired parameters.

If you are creating new job definitions, use:

- **define=yes:** This parameter lets you create or modify a job definition. (If you run EDC data capture without specifying this option, you will only be able to perform data capture based on existing job definitions.) This parameter would typically be used by you, but would be left out when distributed to end users who are simply responsible for entering the data.

This shortcut is now ready to use for defining data capture jobs 8

For users performing data capture:

Both of the following parameters are optional. If not specified, the user will be prompted for the information. Note that if you have multiple users using multiple job definitions, you can easily create a short-cut for each combination of user and job.

- **username="John Doe"** : (Where "John Doe" would be replaced with the desired user name.) You can add this parameter to automatically assign a user with the data capture. Note that names including a space (" ") must be included in quotation marks, otherwise just "John" would be recognized as the user name, and "Doe" would be ignored. This name will appear as part of the data file that is created.

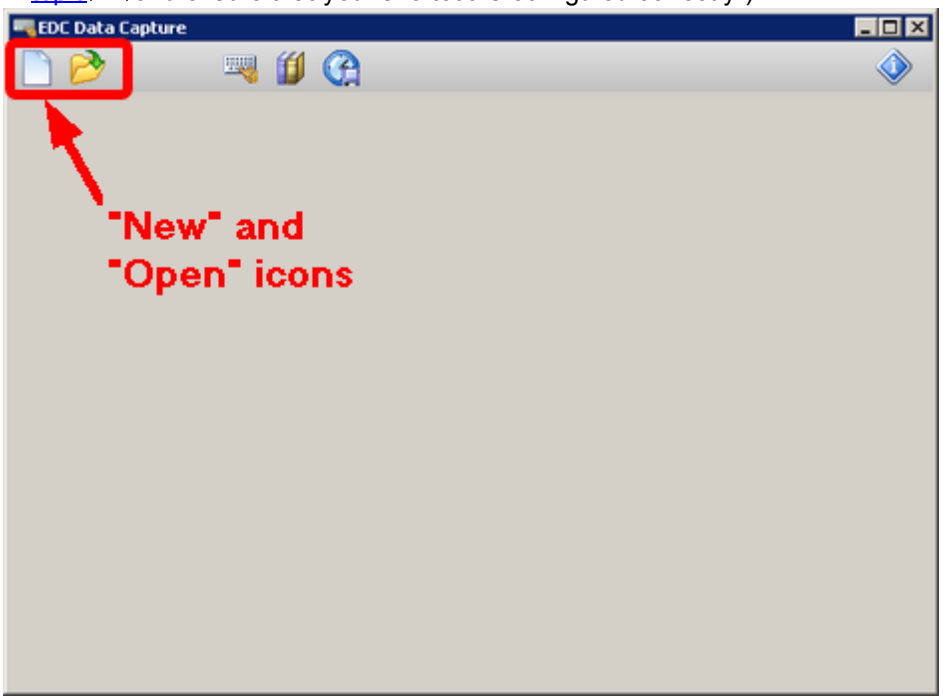
- **capturefile="c:\definition_jobs\mydef.def"** : You can specify the job to be used for data capture sessions started with a shortcut.

© 2010 DocuData Software Corporation

This shortcut is now ready to use for performing data capture 15

3 Defining Data Capture Jobs

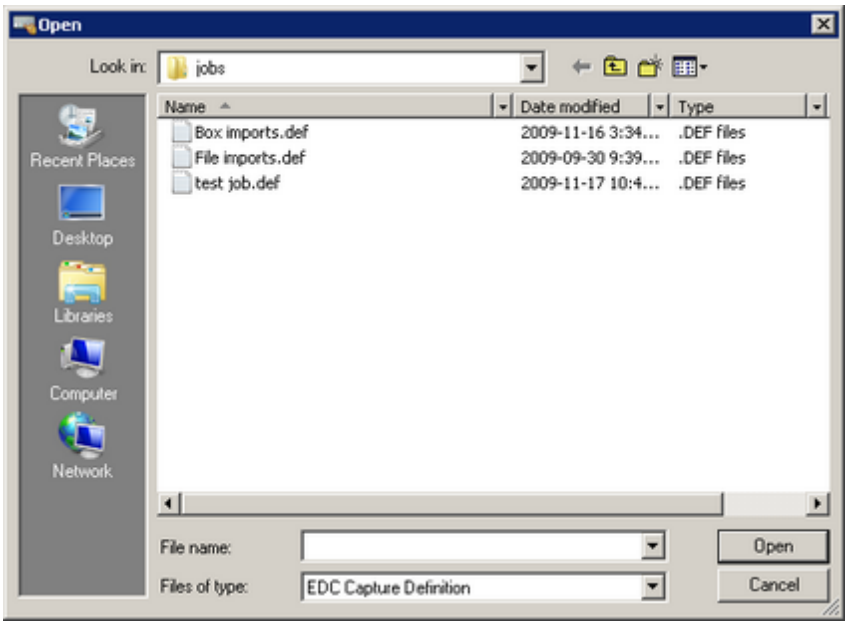
1. To define a data capture job, start EDC Data Capture using a shortcut that includes the command-line parameter "Define=yes". 6 The following window appears. (If icons are missing, consult the previous topic 6 and ensure that your shortcut is configured correctly.)



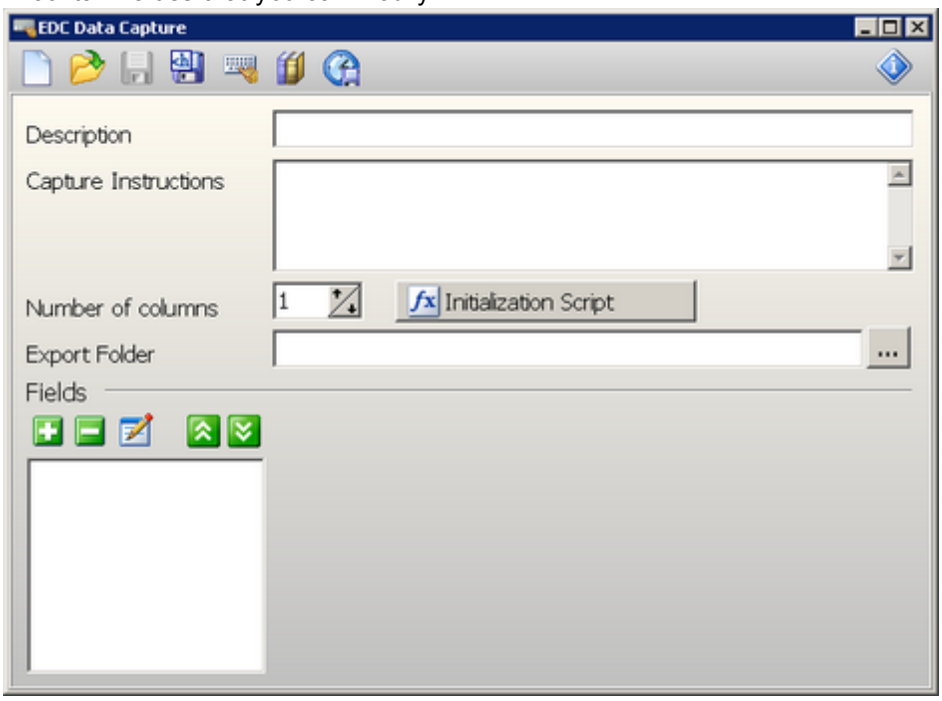
2. To create a new job definition, click the **"new"** icon. Alternatively, to edit an existing job, click the **"open"** icon and browse to the job definition that you wish to modify. Then select the desired job and click "**open**".

Note: Jobs are typically stored in a folder called "Jobs", located inside the "EdcDataCapture" folder, though it is possible to put them elsewhere. They are identified with the extension ".def"

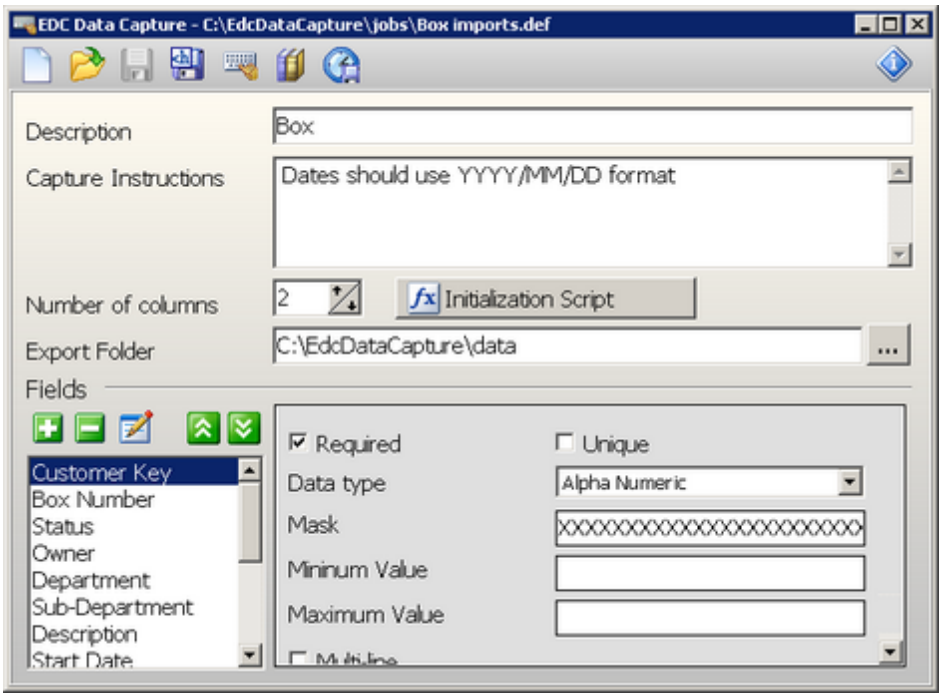
© 2010 DocuData Software Corporation



3. A window will appear. If you are creating a new job, it will be blank. Otherwise, the various fields will contain values that you can modify.



© 2010 DocuData Software Corporation



4. Add a **description**. This text will appear as part of the output file name. For example, if the description is "Box", the date was September 8, 2010, and the user name specified for data capture was "John Doe", the name of the output file would be something like "2010_09_08 19_29_26 Box John Doe_0.csv".

5. Add **capture instructions**.These will be available to the user during data capture, by clicking on the "capture instructions" button which will appear as a dialogue bubble on the right side of the button bar at the top.

These are optional. Instructions will appear in a information window as follows.

6. Specify the **number of columns**. This will determine the layout of the data capture interface for this job. For example, if you specify 2, the data capture window will have a left column and a right one..

7. Optionally, click the "**Initialization Script**" button to add, view, or edit initialization scripts. This is an advanced topic that requires an understanding of programming in the language, "Object Pascal". You are not likely to need to use this button if this is your first time reading this manual.

8. Specify an **Export Folder**, either by typing the path or browsing to it by clicking on the '**...**' button. If you are having data capture performed from within your network, this might be pointing to an accessible folder on your server. If you are distributing it for use on computers not connected to the

© 2010 DocuData Software Corporation

network, you may chose to make the folder a local one such as "**C:\EdcDataCapture\Data**".

9. Define the **fields** to be captured. (see Defining data fields to capture 12 for details).

10. Specify the language to use in data capture by clicking the **Language** button and selecting the desired language from the drop-down list that appears.

11. Save your job by clicking the **Save** or **Save as** button.

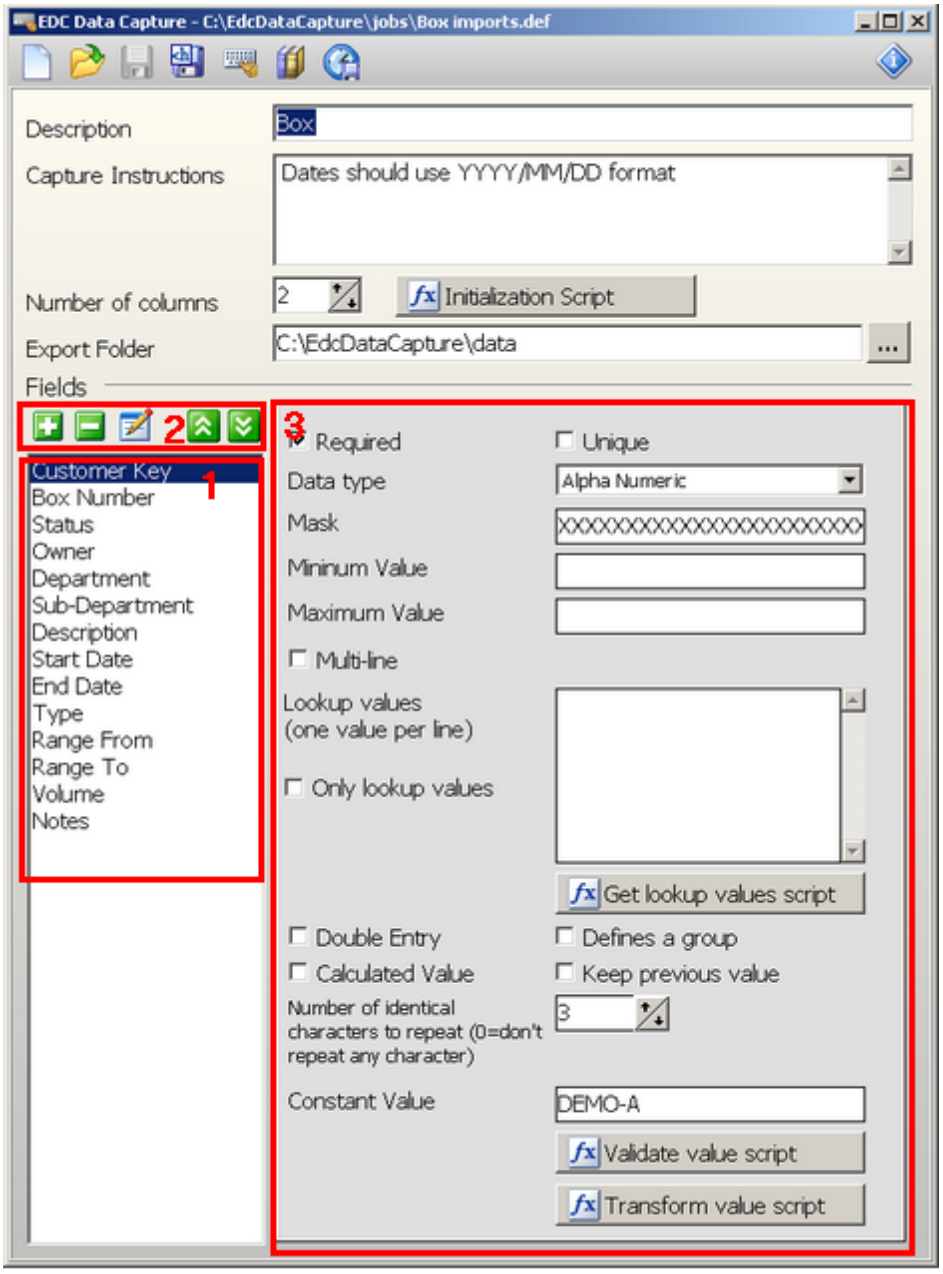
If you are on a network you may have a place where you keep your job definitions. If not, you may wish to save this in a folder such as C:\EdcDataCapture\Jobs, creating this folder if it does not already exist.

Note: If you make changes and attempt to exit without saving your changes, the program will ask you whether you wish to do so.

You are now ready to perform 15 data capture or distribute 14 the data capture job so that others may perform it.

© 2010 DocuData Software Corporation

4 Defining Data Fields to Capture



The lower part of the window is dedicated to the definition of fields for the data capture job.

1. The list of fields appears to the left, with the current field highlighted. (**Box 1**)
2. Above this list, (**Box 2**) buttons enable you to:

- Add a new fields
- Delete the current field
- Rename the current field
- Move the current field up the list
- Move the current field down the list.

© 2010 DocuData Software Corporation

3. To the right (**Box 3**), parameters of the current field are displayed.

To delete a field:
1. Select the field to be deleted by clicking it in **Box 1**.
2. Click the '**-**' button (**Box 2**)
3. Click **Ok** on the message that appears to confirm deletion

To rename a field:
1. Select the field to be renamed by clicking it in **Box 1**.
2. Click the edit button (middle of **Box 2**)
3. Enter a new field name in the box that appears, then click **Ok**

To change the field order:
1. Select the field to be moved by clicking it in **Box 1**.
2. Click the up or down buttons (the two buttons on the right in **Box 2**) to move it up or down.
3. Repeat until you are satisfied with the ordering.

To add a new field:
1. Click the '**+**' button. (**Box 2**)
2. You will be prompted to specify a name for the field. Type one in and press **Enter**.
3. The parameters of the field will be displayed to the right (**Box 3**), and the name of the field will appear in the list (**Box 1**). Complete these as required (see section below, on Field Parameters)

Field Parameters:
- **Required:** When this box is checked, the field will appear in red in the data capture interface, and the user will not be permitted to leave this field blank.
- **Unique:** When this box is checked, a value will be rejected if it already exists for this field within the data.
- **Data type:** Choice field with 4 options:

- *Alpha Numeric* will accept numbers, letters, and various characters according to the mask (specified below), and will treat the values as characters (Eg. 0001 is different than 1)
- *Numeric* fields will treat the value as a number. (eg. 0001 will be stored as 1)
- *Date* fields will accept and store date values
- *Boolean (Yes/No)* fields will accept values of "Y" or "N", and are stored in the spreadsheet as "+" for Y or true and "-" for No or False.

- **Mask:** This will determine what type of values to accept at each position in the field. The nature of the mask will depend on the data type. See the section in the appendix on the Mask 39 for more details.
- **Minimum / Maximum Value:** This parameter is used to specify the minimum and maximum values. Hint: If you are using these fields, it may also be useful to make the fields required if blank values are not accepted.
- **Multi-line:** Check this box if the field should accept multiple lines of text. Note that it only applies if the *data type* parameter is set to *Alpha numeric*. When this is selected, a field will appear to the right, allowing you to set the maximum number of characters that the field can accept.
- **Lookup Values:** If you wish to have a drop-down list that the user can select values for this field, enter the values here, one per line.

-Specify **Only lookup values** by checking the checkbox if users may not type in values that are not in the list.

- Click **Get lookup values script** if you wish to edit the scripts used to create lookup values. (This is an optional, advanced feature. See EDC Data Caputre Scripting Fundamentals 18 for more details.)

- **Double Entry:** With double entry, the field will appear twice on the data entry window. These must contain identical values for the record to be accepted.
- **Defines a group:** With this option, the field will appear in a separate section at the top of the data entry window, and will be excluded from the tab order. (You can get there by clicking with the mouse

© 2010 DocuData Software Corporation

or pressing **Ctrl-G** on the keyboard). This is for fields that are kept the same for many records. (For example, the box number, if you are entering information about files, one box at a time.) The values in this field should be kept from one record to the next.

- **Calculated Value:** Use this option to indicate that the field is automatically calculated using a transformation script. The user will see the value but not be able to modify the field directly during data capture.
- **Keep previous value:** When you go onto the next record, this field will start out filled with the value used for this field in the previous record.
- **Number of identical characters to repeat:** If some but not all of the characters from the previous entry are to be kept, you can enter the number of character to keep here. This is useful if for example you have 8-digit numbers where only the last 2 characters change. The cursor will go to the correct position when you tab to this field in a new record.
- **Constant value:** If the value for this field is always the same for this capture job, then you may set the value here. (For example, if your data output must have the name of the department in it, but this value happens to be the same for all entries that will be created by this job. Compare this to the "Defines a group" option, which the user can modify)
- **Validate value script:** Use this to enter or modify more complex rules to validate that the data for this field is correct. (This is an optional, advanced feature. See EDC Data Caputre Scripting Fundamentals 18 for more details.)
- **Transform value script:** Use this to enter or modify more complex rules to transform the data values entered in this field. Flag the field as "*Calculated Value*". This creates values for the spreadsheet that are based on calculations rather than accepting a value that is typed in or select from a list. (This is an optional, advanced feature. See EDC Data Caputre Scripting Fundamentals 18 for more details.)

5 Distributing and Configuring EDC Data Capture for End Users

1. Open EDC Data Capture on a computer that has internet access. This will allow it to validate the license for off-line use for a period of up to 90 days.

If the application is to be run from a network location

2. Copy the EdcDataCapture folder to a location that is accessible to your users.

If the application is to be run independently

3. Copy the EdcDataCapture folder from the source computer to the destination computer using your favorite mechanism. (The easiest mechanism is to use a USB memory stick. Another option would be to compress the folder and transfer it by FTP or even e-mail.)

Notes:

1- Ensure that the required data capture job definitions are present. These are typically in a sub-folder called **"Jobs"**

2- You may wish to verify that unnecessary files that you may have created are not present. These may include other data capture job definitions (.def), shortcuts for starting data capture, and data files (.csv, typically found in a sub-folder called **"Data"**)

3- You may wish to always install the EdcDataCapture folder directly on the C drive. With this practice it is possible to greatly streamline the configuration.

Configure and test the set-up for your users

4. Create shortcuts for your end users to use in performing data capture. For example, right-click on EdcDataCapture.exe, and select **Send to->Desktop (Create Shortcut)**

5. (*Optional*) Edit this shortcut, adding command line parameters to specify the user name and data

© 2010 DocuData Software Corporation

capture jobs to use (see Creating shortcuts for creating job definitions or performing data capture 6 for detailed instructions)

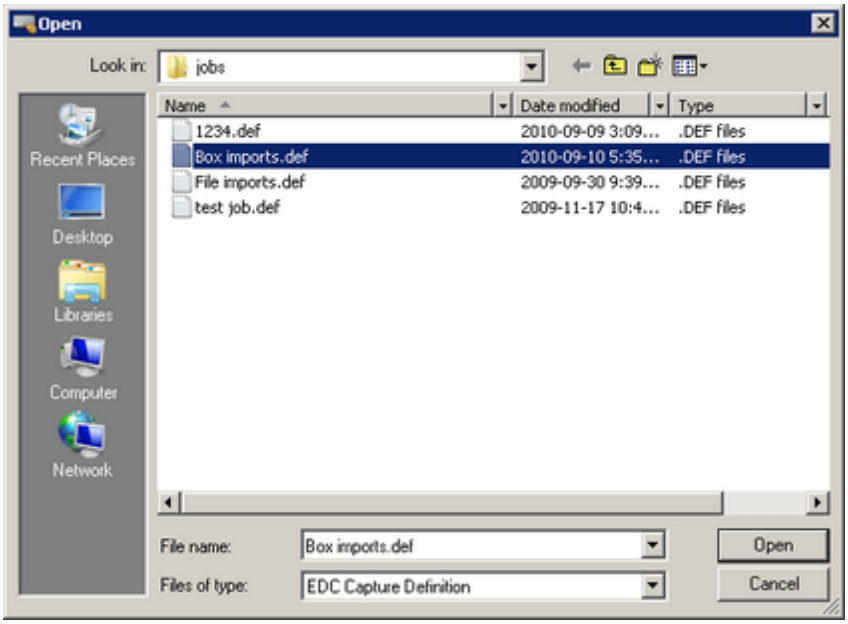
6. Test the shortcut. Perform data capture to ensure that everything is working properly.

Note: The job definition specifies where the data should be saved. You may need to edit the "Export folder" parameter of the job definition 10 for the computer that you are working from.
If you always run the application from the C: drive or from the same network location, then changes will likely be unnecessary.

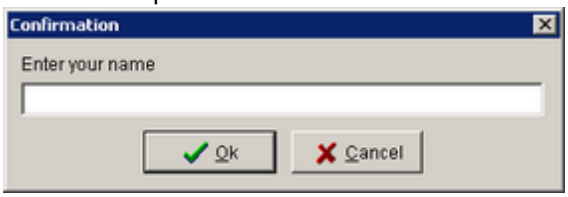
6 Performing Data Capture

Prior to performing data capture, the data capture job should be defined 8.
Depending on the shortcut 6 that you use to start your data capture session, some of the following questions may be omitted.

1. Launch EDC Data Capture in capture mode. This may be done from within EDC Data Capture by clicking on the data capture **icon**, or from outside, by clicking a **shortcut**.
2. If the shortcut did not specify which job to perform, you will be prompted to select one. (These have the extension .def, and are often found in the folder EdcDataCapture\jobs) Select the job and press **Enter**.

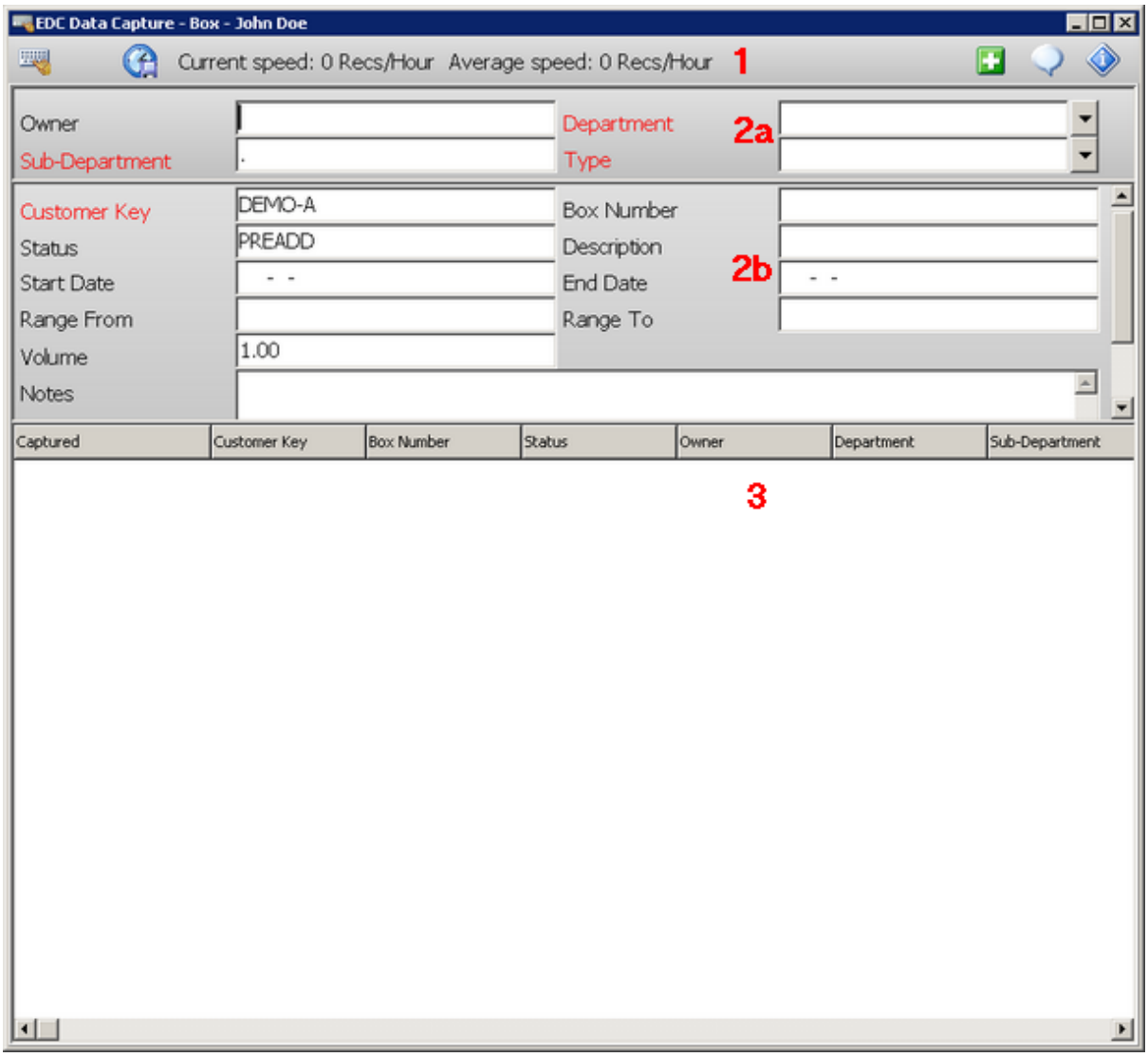


3. If the user name was not specified by the shortcut, you will be prompted to enter one. Type your name and press **Enter**.



4. The data capture interface appears. Enter data, pressing **Ctrl-Enter** after each record.

© 2010 DocuData Software Corporation



- For normal fields, simply type a value and press **Enter** or **Tab** to get to the next field.
- Fields labeled in Red font are "required" fields that may not be left blank
- From within a field with a drop-down list, you may press the **down arrow** on your keyboard or click the button beside a drop-down button to access a list of values to select from.
- Fields above the section divider (region **2a** above) are fields that define groups. Press **Ctrl-G** or use your mouse to access them. They will change frequently than other fields and are not included in the regular tab order.
- Some fields may not allow you to edit them. These are automatically filled in. EDC Data Capture will skip over them.
- Duplicate fields require you to enter the same data twice. This is to protect against typing errors.

1. Button bar

1.1 The **job description** and **user name** are displayed in the title bar
1.2 Click on the "**perform data capture**" button to switch jobs, keeping the same user
1.3 Click here to change the **auto-save frequency**. Note that even if you forget to save, the system will

© 2010 DocuData Software Corporation

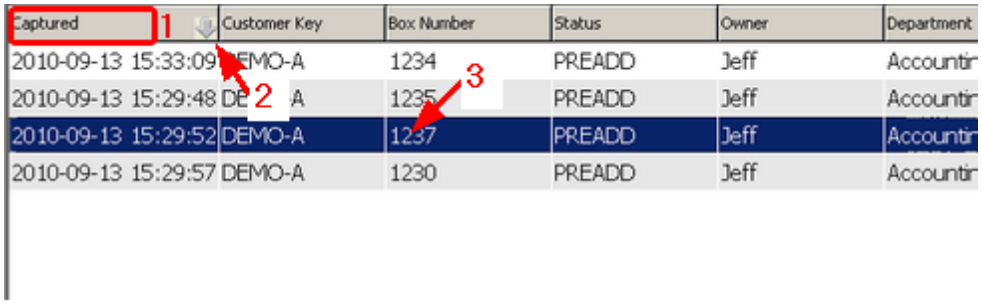
do so automatically when you exit.
1.4 Statistics: the current and average number of records captured per hour are displayed here. The current value is based on the time between the last two records entered.
1.5 Click the '**+**' button to **add** the current data as a record to the data set. (pressing **Ctrl-Enter** or pressing **Enter** in the last field will have the same effect.)
1.6 Click here to view **Capture Instructions** provided as part of the job.
1.7 Click on the About button to view the application version and edit the license 5.

2. Data entry interface
Notice how single-line fields are arranged into columns. The number of columns is specified by the job.

2a Group fields (see the *Defines a group* parameter). Once these have been completed, these values will change less frequently, and will be excluded from the default tab order. To access these fields, press **Ctrl-G**, or use the mouse.

2b Other fields.
- Fields with names in red are required fields
- If a field appears twice, it is necessary to enter the same value twice to ensure accuracy
- Fields with constant or calculated values are skipped by the tab order and will be filled automatically.
- Drop-down lists allow you to select from a list of values. Press the down arrow from within the field or click the drop-down button.

3. Data



Each record captured during the current session will be displayed in the data pane. You may double-click on the record to go to that record to edit it. (Note that data from old sessions will not be displayed and cannot be modified in this way.
3.1 The *Captured* column is automatically added by EDC Data Capture. It records the time at which the record was saved or most recently edited.
3.2 Click a column header to sort by it; click it again to reverse the sort order
3.3 Double-click a row to display it for editing, and Ctrl-E to save it again

7 Retrieving Captured Data

You can find the captured data in the folder specified by the job definition 8.
(In many cases, this will be something like C:\EdcDataCapture\Data.)

Each data capture session creates a separate spreadsheet with the extension ".csv".

The file name consists of: [Date in YYYY_MM_DD_HH_MM format] [Job name] [User Name]_0.csv.
The final number is usually 0, but will change if two otherwise identical file names are create.

© 2010 DocuData Software Corporation

8 Using Scripts to Enhance Your Data Capture

While EDC Data Capture can be very valuable without scripts, the ability to run scripts makes it far more powerful.

Scripts can:
- Retrieve data from external data sources
- Retrieve or create lists of values for drop-down lists on fields
- Perform complex validations, including validations of one field based on others
- Transform the values in a field
- Populate a field with a calculated value.

If you require custom scripting, having DocuData do this for you can be a very cost-effective option. (Contact support@docudatasoft.com for an estimate.)

Additionally, if you are comfortable with programming and willing to experiment, the examples 18 and reference material 21 in this manual can help you to do your own automation.

If this interests you, we recommend trying out the examples in this chapter then adapting them with the help of the API 21. Additional documentation for the various libraries can be found from other sources online.

Note: This chapter is intended for advanced users. It contains examples and hint rather than complete instructions.

8.1 EDC Data Capture Scripting Fundamentals

The following series of examples illustrate the most common types of scripts used with EDC Data Capture. A full course in programming is beyond the scope of this manual. However, with the help of the EDC Data Capture API Reference 21, you may be able to adapt the following examples to meet your needs.

Examples of scripts for getting look-up values 18
Examples of scripts for validating values 19
Examples of scripts for transforming values 20

8.1.1 Examples of scripts for getting look-up values

Populate look-up list with static values:

```
Pasting the following into "get look-up values script" for a field will add
the values 'Ms', 'Mrs', 'Mr', 'Dr', and 'Sr' to it as a drop down list.
Note that if values are added through the job description, these will be
added to the end of the list.

function GetLookupValues( const GlobalData : Variant ) : Variant;
begin
    Result := VarArrayCreate( [ 1, 5 ], varVariant );

    VarArrayPut( Result, 'Ms', [1]);
    VarArrayPut( Result, 'Mrs', [2]);
    VarArrayPut( Result, 'Mr', [3]);
    VarArrayPut( Result, 'Dr', [4]);
```

© 2010 DocuData Software Corporation

```
    VarArrayPut( Result, 'Sr', [5]);
end;
begin
end.
```

Populate look-up list with main department names from EDC RC:

This is a very practical and useful example.

You may use the following code to populate the list with main department names, taken from a database with the DSN, 'EDC_RC', for the customer with custno = 112. (This is customer DEMO-D from the EDC RC demo.)

In general, you will need the ODBC data source name (DSN) of the database, and a valid SQL statement that will produce the list that you want. You may also require login credentials for the data source, but this example does not require them.

```
function GetLookupValues( const GlobalData : Variant ) : Variant;
var
    SQLResult : Variant;
    Row       : Variant;
    RowNo     : Integer;
    ColumnNo  : Integer;
begin
     SQLResult := ExecuteQuery( 'EDC_RC', '', '', 'select MainDeptName from
Department where CustNo = 112' );

     if VarIsNull( SQLResult ) then
     begin
         MessageBox( 'There are no departments returned from query of
Department field');
         Result := NULL;
     end
     else
     begin
         Result := VarArrayCreate( [ 1, VarArrayHighBound(SQLResult, 1) ],
varVariant );

         for RowNo := 1 to VarArrayHighBound(SQLResult, 1) do
         begin
             Row := VarArrayGet( SQLResult, [RowNo] )

             VarArrayPut( Result, VarArrayGet( Row, [1]), [RowNo]);
         end;
     end;
end;
begin
end.
```

8.1.2 Examples of scripts for validating values

Here is an example where the Age field is being looked at to validate another field.

© 2010 DocuData Software Corporation

```
+++

To use this script example, make a job definition with a field called "Age"
and position it as the third. To perform a similar validation for a
different field at a different position, replace "3" and "'Age'" with the
appropriate values in FieldValueByIndex( FieldNames, FieldValues, 3,
'Age' )

function ValidateValue( const Params : Variant ) : Variant;
var
    GlobalData   : Variant;
    FieldValue   : Variant;
    FieldNames   : Variant;
    FieldValues  : Variant;
    Age          : Variant;

begin
    GlobalData  := VarArrayGet( Params, [1] );
    FieldValue  := VarArrayGet( Params, [2] );
    FieldNames  := VarArrayGet( Params, [3] );
    FieldValues := VarArrayGet( Params, [4] );

    { age is the third defined field }
    Age := FieldValueByIndex( FieldNames, FieldValues, 3, 'Age' );

    if Age < 18 then
    begin
        Result := VarArrayCreate( [ 1, 2 ], varVariant );
        VarArrayPut( Result, False, [1]);
        VarArrayPut( Result, 'Under 18, cannot drink', [2]);
    end
    else
    begin
        Result := VarArrayCreate( [ 1, 1 ], varVariant );
        VarArrayPut( Result, True, [1]);
    end
end;
begin
end.
```

8.1.3 Examples of scripts for transforming values

The following example takes the input value for a field and stores it in upper case. (Note that this is and example, and that this can also be done with a mask.)

Transform the entered text to uppercase and store it in the same field

To apply this transformation to a field, go to the field and click on the "**Transform value script**" button to paste in the following:

```
function TransformValue( const Params : Variant ) : Variant;
var
    GlobalData   : Variant;
    FieldValue   : Variant;
```

© 2010 DocuData Software Corporation

```
    FieldNames   : Variant;
    FieldValues  : Variant;

begin
    GlobalData  := VarArrayGet( Params, [1] );
    FieldValue  := VarArrayGet( Params, [2] );
    FieldNames  := VarArrayGet( Params, [3] );
    FieldValues := VarArrayGet( Params, [4] );

    Result := StrUpper(FieldValue);

end;
begin
end.
```

Calculate the current field from a value entered in another field

For this example, you will require the 5th field to be called 'Max Heart Rate' and contain a numeric value. The script should be applied to a different field, which will also be a numeric field, and will be flagged as "Calculated Value"

To apply this transformation to a field, go to the field and click on the "**Transform value script**" button to paste in the following:

```
function TransformValue( const Params : Variant ) : Variant;
var
    GlobalData   : Variant;
    FieldValue   : Variant;
    FieldNames   : Variant;
    FieldValues  : Variant;
    HeartRate    : Variant;

begin
    GlobalData  := VarArrayGet( Params, [1] );
    FieldValue  := VarArrayGet( Params, [2] );
    FieldNames  := VarArrayGet( Params, [3] );
    FieldValues := VarArrayGet( Params, [4] );

    HeartRate := FieldValueByIndex( FieldNames, FieldValues, 5, 'Max Heart
Rate' );

    Result := HeartRate * 10;
end;
begin
end.
```

8.2 EDC Data Capture API Reference

Here is the list of types, constants & functions (API) that the EDC Data Capture software makes available to you when scripting. Some come from DocuData while others come from Object Pascal (Delphi) and some from the Jedi library.

DocuData API

© 2010 DocuData Software Corporation

Functions & Procedures

```
CloseAllProcesses(const FileName:String; const MatchFileNameOnly:Boolean):
Boolean

ForceDelete(const FileName:String):Boolean

RaiseIf(const Condition:Boolean; const Msg:String)

RaiseIfNot(const Condition:Boolean; const Msg:String)

ExecNoWait(const CmdLine:String):Boolean

ExecWait(const CmdLine:String; var ExitCode:LongInt):Boolean

ExecNoWaitInDir(const CmdLine:String; const Directory:String):Boolean

ExecWaitInDir(const CmdLine:String; const Directory:String; var ExitCode:
LongInt):Boolean

MessageBox(const Msg:String)

InputQuery(const ACaption, APrompt:string; var Value:string):Boolean

ParamCount:Integer

ParamStr(const Index:Integer):String

SendKeys(const WindowName:String; const Keys:String)

DelayMS(const MilliSeconds:LongInt)

PathAddSeparator(const Path:string):string

PathGetShortName(const Path:string):string

CreateEmptyFile(const FileName:string)

DelTree(const Path:string):Boolean

DirectoryExists(const Name:string):Boolean

FileExists(const FileName:string):Boolean

FileGetSize(const FileName:string):Integer

FileGetTempName(const Prefix:string):string

ForceDirectories(Name:string):Boolean

GetSizeOfFile(const FileName:string):Int64

IsDirectory(const FileName:string):Boolean
```

© 2010 DocuData Software Corporation

```
CurrentDirectory:String

Today:LongInt

CurrentTime:LongInt

DateToDateString(Picture:String; Julian:LongInt):String

DateStringToDate(Picture, S:String):LongInt

TimeToTimeString(Picture:String; T:LongInt):String

TimeStringToTime(Picture, S:String):LongInt

ValidDate(Day, Month, Year:Integer):Boolean

DMYtoDate(Day, Month, Year:Integer):LongInt

DateToDMY(Julian:LongInt; var Day, Month, Year:Integer)

IncDate(Julian:LongInt; Days, Months, Years:Integer):LongInt

DateDiff(Date1, Date2:LongInt; var Days, Months, Years:Integer)

DayOfWeek(Julian:LongInt):Integer

IsLeapYear(Year:Integer):Boolean

DaysInMonth(Month, Year:Integer):Integer

ValidTime(Hours, Minutes, Seconds:Integer):Boolean

TimeToHMS(T:LongInt; var Hours, Minutes, Seconds:Byte)

HMStoTime(Hours, Minutes, Seconds:Byte):LongInt

TimeDiff(Time1, Time2:LongInt; var Hours, Minutes, Seconds:Byte)

IncTime(T:LongInt; Hours, Minutes, Seconds:Byte):LongInt

DecTime(T:LongInt; Hours, Minutes, Seconds:Byte):LongInt

CreateTempDir(var OutDir:String; const BaseDir:String):Boolean

GetTempDir:string;

DeleteFiles(const FileMask:string):Boolean

MoveFile(const FileName, DestName:String)

CopyFile(const Source:String; const Dest:String):Boolean
```

© 2010 DocuData Software Corporation

```
DeleteFile(const FileName:String):Boolean

CopyFiles(const SourceDir:String; const DestDir:String; const Mask:String)

ShortBeep

LongBeep

ExtractFilePath(const FullFileName:String):String

ExtractFileExt(const FullFileName:String):String

ExtractFileNameWithExt(const FullFileName:String):String

ExtractFileNameWithoutExt(const FullFileName:String):String

ChangeFileExtension(const FileName:String; const NewExt:String):String

AddFileNameToPath(const FileName:String;const DirName:String):String

ChangeDirectory(const Directory:String):Boolean

ExpandFileName(const FileName:string):string

IsReadOnly(const FileName:String):Boolean

SetReadOnly(const FileName:String):Boolean

ClearReadOnly(const FileName:String):Boolean

GetFileDateTime(const FileName:String; var Date:LongInt; var Time:LongInt):
Boolean

SetFileDateTime(const FileName:String; const Date:LongInt; const Time:
LongInt):Boolean

BuildFileList(const Path:string; const List:TStrings;const Recurse:Boolean)

SearchAndReplaceInFile(const FileName:String; const SearchFor:String; const
ReplaceWith:String)

GetEnvString(const Entry:String):String

WordCount(const S:string; const WordDelims:TSysCharSet):Integer

WordPosition(const N:Integer; const S:string; const WordDelims:
TSysCharSet):Integer

ExtractWord(N:Integer; const S:string; const WordDelims:TSysCharSet):string

WaitForWindow(const Caption:String; const TimeOutMS:LongInt)

AddLogMessage(const LogFileName:String; const Msg:String)
```

© 2010 DocuData Software Corporation

```
ReadStruct(const XMLData:String):TXrStruct

ExecuteQuery(const DSN:String; const UserName:String; const Password:
String; const SelectSQL:String):Variant

FieldValueByIndex(const FieldNames:Variant; const FieldValues:Variant;
const Index:Integer; const FieldName:String):Variant
```

© 2010 DocuData Software Corporation

XML-RPC classes:

```
-------------------------------------------------------------------------------------------------------------------------
TXrValue = class(TObject)
public
   class function Noun : AnsiString; virtual; abstract;
   function AsXML : AnsiString; virtual; abstract;
end;
-------------------------------------------------------------------------------------------------------------------------

-------------------------------------------------------------------------------------------------------------------------
TXrArray = class(TXrValue)
protected
   FArray : TArray;

   function GetSize : TArrayIndex;
   function GetItem( const Index : TArrayIndex ) : TXrValue;
   function GetOwnership : Boolean;
   procedure SetOwnership( const Value : Boolean );
public
   class function Noun : AnsiString; override;

   constructor Create;
   destructor Destroy; override;

   function AsXML : AnsiString; override;

   procedure Insert( const Value : TXrValue );
   procedure Delete( const Index : TArrayIndex );
   procedure Clear;
   procedure InsertStr( const S : AnsiString );
   procedure InsertInt( const I : LongInt );
   procedure InsertBool( const B : Boolean );
   procedure InsertFloat( const F : Double );
   procedure InsertDateTime( const Date : TStDate; const Time : TStTime );

   function StrValue( const Index : TArrayIndex ) : AnsiString;
   function IntValue( const Index : TArrayIndex ) : LongInt;
   function BoolValue( const Index : TArrayIndex ) : Boolean;
   function FloatValue( const Index : TArrayIndex ) : Double;
   function DateValue( const Index : TArrayIndex ) : TStDate;
   function TimeValue( const Index : TArrayIndex ) : TStTime;
   function DateTimeValue( const Index : TArrayIndex ) : TStDateTimeRec;

   property Size : TArrayIndex read GetSize;
   property Items[ const Index : TArrayIndex ] : TXrValue read GetItem;
default;
   property Ownership : Boolean read GetOwnership write SetOwnership;
end;
-------------------------------------------------------------------------------------------------------------------------

-------------------------------------------------------------------------------------------------------------------------
TXrStruct = class(TXrValue)
```

© 2010 DocuData Software Corporation

```
protected
   FTable : THashTable;
   FKeys  : TStringList;

   function GetValueByKey( const Key : AnsiString ) : TXrValue;
   function GetSize : TArrayIndex;
   function GetKey( const Index : TArrayIndex ) : AnsiString;
public
   class function Noun : AnsiString; override;

   constructor Create;
   destructor Destroy; override;

   function AsXML : AnsiString; override;

   procedure Insert( const Key : AnsiString; const Value : TXrValue );
   procedure InsertStr( const Key : AnsiString; const S : AnsiString );
   procedure InsertInt( const Key : AnsiString; const I : LongInt );
   procedure InsertBool( const Key : AnsiString; const B : Boolean );
   procedure InsertFloat( const Key : AnsiString; const F : Double );
   procedure InsertDateTime( const Key : AnsiString; const Date : TStDate;
const Time : TStTime );

   { ValuesByKey returns nil if Key not found }
   property ValuesByKey[ const Key : AnsiString ] : TXrValue read
GetValueByKey; default;

   property Size : TArrayIndex read GetSize;

   { Simple retrieval methods. Will raise exception if key if not of the
right type }
   function StrValue( const Key : AnsiString ) : AnsiString;
   function IntValue( const Key : AnsiString ) : LongInt;
   function BoolValue( const Key : AnsiString ) : Boolean;
   function FloatValue( const Key : AnsiString ) : Double;
   function DateValue( const Key : AnsiString ) : TStDate;
   function TimeValue( const Key : AnsiString ) : TStTime;
   function DateTimeValue( const Key : AnsiString ) : TStDateTimeRec;

   function StructValue( const Key : AnsiString ) : TXrStruct;

   function ArrayValue( const Key : AnsiString ) : TXrArray;

   property Keys[ const Index : TArrayIndex ] : AnsiString read GetKey;

   procedure DeleteKey( const Key : AnsiString );
end;
-------------------------------------------------------------------------------------------------------------------------------

-------------------------------------------------------------------------------------------------------------------------------
TXrSimple = class(TXrValue)
public
   constructor CreateFromContent( const Content : AnsiString ); virtual;
```

© 2010 DocuData Software Corporation

```
abstract;
end;
-------------------------------------------------------------------------------------------------------------------------------

-------------------------------------------------------------------------------------------------------------------------------
TXrInt4 = class(TXrSimple)
protected
   FValue : LongInt;
public
   class function Noun : AnsiString; override;

   constructor Create( const AValue : LongInt );
   constructor CreateFromContent( const Content : AnsiString ); override;

   function AsXML : AnsiString; override;

   property Value : LongInt read FValue write FValue;
end;
-------------------------------------------------------------------------------------------------------------------------------

-------------------------------------------------------------------------------------------------------------------------------
TXrBoolean = class(TXrSimple)
protected
   FValue : Boolean;
public
   class function Noun : AnsiString; override;

   constructor Create( const AValue : Boolean );
   constructor CreateFromContent( const Content : AnsiString ); override;

   function AsXML : AnsiString; override;

   property Value : Boolean read FValue write FValue;
end;
-------------------------------------------------------------------------------------------------------------------------------

-------------------------------------------------------------------------------------------------------------------------------
TXrString = class(TXrSimple)
protected
   FValue : AnsiString;
public
   class function Noun : AnsiString; override;

   constructor Create( const AValue : AnsiString );
   constructor CreateFromContent( const Content : AnsiString ); override;

   function AsXML : AnsiString; override;

   property Value : AnsiString read FValue write FValue;
end;
-------------------------------------------------------------------------------------------------------------------------------

-------------------------------------------------------------------------------------------------------------------------------
```

© 2010 DocuData Software Corporation

```
TXrDouble = class(TXrSimple)
protected
   FValue : Double;
public
   class function Noun : AnsiString; override;

   constructor Create( const AValue : Double );
   constructor CreateFromContent( const Content : AnsiString ); override;

   function AsXML : AnsiString; override;

   property Value : Double read FValue;
end;
```

```
TXrDateTime = class(TXrSimple)
protected
   FDateTimeRec : TStDateTimeRec;
   function GetDate : TStDate;
   function GetTime : TStTime;
   procedure SetDate( const D : TStDate );
   procedure SetTime( const T : TStTime );
public
   class function Noun : AnsiString; override;

   constructor Create( const ADate : TStDate; const ATime : TStTime );
   constructor CreateFromContent( const Content : AnsiString ); override;

   function AsXML : AnsiString; override;

   property Date : TStDate read GetDate write SetDate;
   property Time : TStTime read GetTime write SetTime;
   property DateTime : TStDateTimeRec read FDateTimeRec;
end;
```

```
TXrBinary = class(TXrSimple)
protected
   FData : AnsiString;
public
   class function Noun : AnsiString; override;

   constructor Create( const Buffer; const Size : LongInt );
   constructor CreateFromContent( const Content : AnsiString ); override;
   destructor Destroy; override;

   function AsXML : AnsiString; override;

   procedure SetData( var OutData );

   property Data : AnsiString read FData;
```

© 2010 DocuData Software Corporation

```
end;
```

© 2010 DocuData Software Corporation

Object Pascal API

```
VarIsNull(const V: Variant): Boolean

VarToStr(const V: Variant): string

VarSameValue(const A, B: Variant): Boolean

VarClear(var V : Variant)

VarIsArray(const A: Variant): Boolean

VarArrayCreate(const Bounds: array of Integer; AVarType: TVarType): Variant

VarArrayGet(const A: Variant; const Indices: array of Integer): Variant

VarArrayPut(var A: Variant; const Value: Variant; const Indices: array of
Integer)

VarArrayDimCount(const A: Variant): Integer

VarArrayLowBound(const A: Variant; Dim: Integer): Integer

VarArrayHighBound(const A: Variant; Dim: Integer): Integer
```

Delphi's standard classes:

```
TMemoryStream
TFileStream
TStringList
TDateTime
```

Delphi standard date routines:

```
EncodeDate(Year, Month, Day: Word): TDateTime

EncodeTime(Hour, Min, Sec, MSec: Word): TDateTime

TryEncodeDate(Year, Month, Day: Word; var Date: TDateTime): Boolean

TryEncodeTime(Hour, Min, Sec, MSec: Word; var Time: TDateTime): Boolean

DecodeDate(const DateTime: TDateTime; var Year, Month, Day: Word)

DecodeTime(const DateTime: TDateTime; var Hour, Min, Sec, MSec: Word)

DayOfWeek(const DateTime: TDateTime): Word

Date: TDateTime

Time: TDateTime
```

© 2010 DocuData Software Corporation

```
Now: TDateTime

DateTimeToUnix(D: TDateTime): Int64

UnixToDateTime(U: Int64): TDateTime

DateToStr(D: TDateTime): string

StrToDate(const s: string): TDateTime

FormatDateTime(const fmt: string; D: TDateTime): string
```

© 2010 DocuData Software Corporation

Jedi Library API

Types

TSysCharSet
TReplaceFlags
Float

Constants

AnsiNull
AnsiSoh
AnsiStx
AnsiEtx
AnsiEot
AnsiEnq
AnsiAck
AnsiBell
AnsiBackspace
AnsiTab
AnsiLF
AnsiVerticalTab
AnsiFormFeed
AnsiCR
AnsiSo
AnsiSi
AnsiDle
AnsiDc1
AnsiDc2
AnsiDc3
AnsiDc4
AnsiNak
AnsiSyn
AnsiEtb
AnsiCan
AnsiEm
AnsiEndOfFile
AnsiEscape
AnsiFs
AnsiGs
AnsiRs
AnsiUs
AnsiSpace
AnsiComma
AnsiBackslash
AnsiForwardSlash
AnsiDoubleQuote
AnsiSingleQuote
AnsiWhiteSpace
AnsiTab
AnsiLineFeed
AnsiVerticalTab

© 2010 DocuData Software Corporation

```
AnsiFormFeed
AnsiCarriageReturn
AnsiSpace
C1_UPPER
C1_LOWER
C1_DIGIT
C1_SPACE
C1_PUNCT
C1_CNTRL
C1_BLANK
C1_XDIGIT
C1_ALPHA
```

Functions & Procedures

```
StrIsAlpha( const S : string) : Boolean

StrIsAlphaNum( const S : string) : Boolean

StrIsAlphaNumUnderscore( const S : string) : Boolean

StrContainsChars( const S : string; Chars : TSysCharSet; CheckAll :
Boolean) : Boolean

StrConsistsOfNumberChars( const S : string) : Boolean

StrIsDigit( const S : string) : Boolean

StrIsSubset( const S : string; const ValidChars : TSysCharSet) : Boolean

StrSame( const S1, S2 : string) : Boolean

StrCenter( const S : string; L : Integer; C : Char) : string

StrCharPosLower( const S : string; CharPos : Integer) : string

StrCharPosUpper( const S : string; CharPos : Integer) : string

StrDoubleQuote( const S : string) : string

StrEnsureNoPrefix( const Prefix, Text : string) : string

StrEnsureNoSuffix( const Suffix, Text : string) : string

StrEnsurePrefix( const Prefix, Text : string) : string

StrEnsureSuffix( const Suffix, Text : string) : string

StrEscapedToString( const S : string) : string

StrLower( const S : string) : string

StrLowerInPlace( var S : string)
```

© 2010 DocuData Software Corporation

```
StrMove( var Dest : string; const Source : string; const ToIndex,
FromIndex, Count : Integer)

StrPadLeft( const S : string; Len : Integer; C : Char) : string

StrPadRight( const S : string; Len : Integer; C : Char) : string

StrProper( const S : string) : string

StrQuote( const S : string; C : Char) : string

StrRemoveChars( const S : string; const Chars : TSysCharSet) : string

StrKeepChars( const S : string; const Chars : TSysCharSet) : string

StrReplace( var S : string; const Search, Replace : string; Flags :
TReplaceFlags)

StrReplaceChar( const S : string; const Source, Replace : Char) : string

StrReplaceChars( const S : string; const Chars : TSysCharSet; Replace :
Char) : string

StrReplaceButChars( const S : string; const Chars : TSysCharSet; Replace :
Char) : string

StrRepeat( const S : string; Count : Integer) : string

StrRepeatLength( const S : string; L : Integer) : string

StrReverse( const S : string) : string

StrReverseInPlace( var S : string)

StrSingleQuote( const S : string) : string

StrSmartCase( const S : string; Delimiters : TSysCharSet) : string

StrStringToEscaped( const S : string) : string

StrStripNonNumberChars( const S : string) : string

StrToHex( const Source : string) : string

StrTrimCharLeft( const S : string; C : Char) : string

StrTrimCharsLeft( const S : string; const Chars : TSysCharSet) : string

StrTrimCharRight( const S : string; C : Char) : string

StrTrimCharsRight( const S : string; const Chars : TSysCharSet) : string
```

© 2010 DocuData Software Corporation

```
StrTrimQuotes( const S : string) : string

StrUpper( const S : string) : string

StrUpperInPlace( var S : string)

StrOemToAnsi( const S : string) : string

StrAnsiToOem( const S : string) : string

StrAllocSize( const S : string) : Longint

StrLength( const S : string) : Longint

StrResetLength( var S : string)

StrCharCount( const S : string; C : Char) : Integer

StrCharsCount( const S : string; Chars : TSysCharSet) : Integer

StrStrCount( const S, SubS : string) : Integer

StrCompare( const S1, S2 : string) : Integer

StrCompareRange( const S1, S2 : string; const Index, Count : Integer) :
Integer

StrFillChar( const C : Char; Count : Integer) : string

StrFind( const Substr, S : string; const Index : Integer) : Integer

StrHasPrefix( const S : string; const Prefixes : array of string) : Boolean

StrIndex( const S : string; const List : array of string) : Integer

StrILastPos( const SubStr, S : string) : Integer

StrIPos( const SubStr, S : string) : Integer

StrIsOneOf( const S : string; const List : array of string) : Boolean

StrLastPos( const SubStr, S : string) : Integer

StrMatch( const Substr, S : string; const Index : Integer) : Integer

StrMatches( const Substr, S : string; const Index : Integer) : Boolean

StrNIPos( const S, SubStr : string; N : Integer) : Integer

StrNPos( const S, SubStr : string; N : Integer) : Integer

StrPrefixIndex( const S : string; const Prefixes : array of string) :
Integer
```

© 2010 DocuData Software Corporation

```
StrSearch( const Substr, S : string; const Index : Integer) : Integer

StrAfter( const SubStr, S : string) : string

StrBefore( const SubStr, S : string) : string

StrBetween( const S : string; const Start, Stop : Char) : string

StrChopRight( const S : string; N : Integer) : string

StrLeft( const S : string; Count : Integer) : string

StrMid( const S : string; Start, Count : Integer) : string

StrRestOf( const S : string; N : Integer) : string

StrRight( const S : string; Count : Integer) : string

CharEqualNoCase( const C1, C2 : Char) : Boolean

CharIsAlpha( const C : Char) : Boolean

CharIsAlphaNum( const C : Char) : Boolean

CharIsBlank( const C : Char) : Boolean

CharIsControl( const C : Char) : Boolean

CharIsDelete( const C : Char) : Boolean

CharIsDigit( const C : Char) : Boolean

CharIsLower( const C : Char) : Boolean

CharIsNumberChar( const C : Char) : Boolean

CharIsPrintable( const C : Char) : Boolean

CharIsPunctuation( const C : Char) : Boolean

CharIsReturn( const C : Char) : Boolean

CharIsSpace( const C : Char) : Boolean

CharIsUpper( const C : Char) : Boolean

CharIsWhiteSpace( const C : Char) : Boolean

CharType( const C : Char) : Word

CharHex( const C : Char) : Byte
```

© 2010 DocuData Software Corporation

```
CharLower( const C : Char) : Char

CharUpper( const C : Char) : Char

CharToggleCase( const C : Char) : Char

CharPos( const S : string; const C : Char; const Index : Integer) : Integer

CharLastPos( const S : string; const C : Char; const Index : Integer) :
Integer

CharIPos( const S : string; C : Char; const Index : Integer) : Integer

CharReplace( var S : string; const Search, Replace : Char) : Integer

StrIToStrings( S, Sep : string; const List : TStrings; const
AllowEmptyString : Boolean)

StrToStrings( S, Sep : string; const List : TStrings; const
AllowEmptyString : Boolean)

StringsToStr(const List:TStrings; const Sep:string; const AllowEmptyString:
Boolean) : string

TrimStrings( const List : TStrings; DeleteIfEmpty : Boolean)

TrimStringsRight( const List : TStrings; DeleteIfEmpty : Boolean)

TrimStringsLeft( const List : TStrings; DeleteIfEmpty : Boolean)

AddStringToStrings( const S : string; Strings : TStrings; const Unique :
Boolean) : Boolean

BooleanToStr( B : Boolean) : string

FileToString( const FileName : string) : String

StringToFile( const FileName : string; const Contents : String)

StrToken( var S : string; Separator : Char) : string

StrTokens( const S : string; const List : TStrings)

StrTokenToStrings( S : string; Separator : Char; const List : TStrings)

StrToFloatSafe( const S : string) : Float

StrToIntSafe( const S : string) : Integer

StrNormIndex( const StrLen : Integer; var Index : Integer; var Count :
Integer)
```

© 2010 DocuData Software Corporation

9 Appendix

9.1 Mask values

The following characters may be used in creating masks to determine what values a field can accepts as input. .

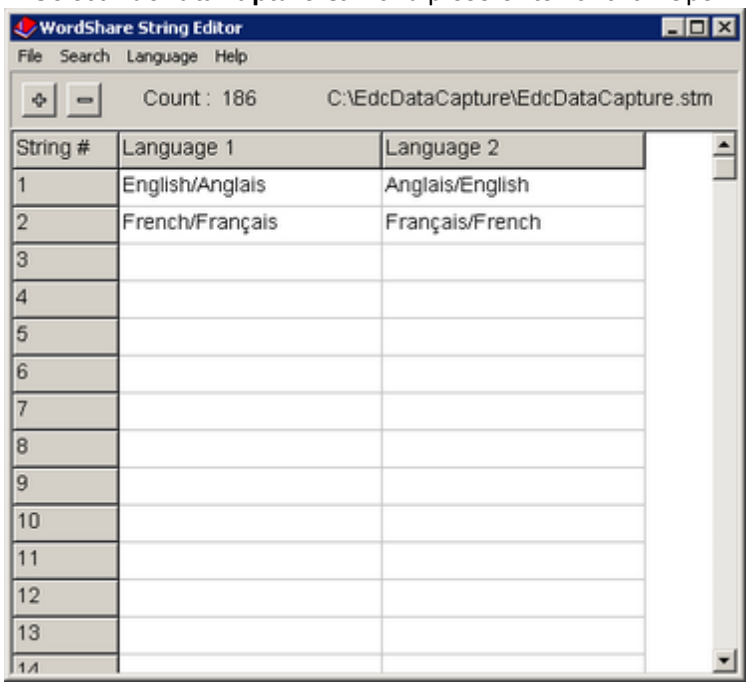
- "X" any character
- "!" any character, forced to upper case
- "L" any character, forced to lower case
- "x" any character, first letter of a word forced to upper case
- "a" alphas and "-" and "," and "."
- "A" alphas and "-" and "," and ".", forced to upper case
- "l" alphas and "-" and "," and ".", forced to lower case
- "1" numbers only, no spaces
- "9" numbers and spaces only
- "i" numbers and spaces and "-"
- "#" numbers and spaces and "-" and "."

You may also enter specific characters. For example, (999) 999-9999 will require that the data have the "(", ")", and "-" characters at the specified positions.

9.2 Editing the interface wording

You may edit the terminology used in the EDC Data Capture application as follows:

1. From the same folder as EdcDataCapture.exe, open **WsEdit32.exe**.
2. From the menu, select **File->Open**.
3. Select **EdcDataCapture.stm** and press **enter** or click Open. The dictionary is opened as follows:



© 2010 DocuData Software Corporation

To change a string
- Go to the string, then type or paste over the existing text.

To add another language from an existing file
- From the menu select **File->Import**.
- Select which string number to import from the source file. This will be added as the next language in the existing file.

To add a new language
- From the menu select **Language->Add**. You will be prompted to select which language to copy from.

To search for an exact string in the dictionary
- Press **Ctrl**-**F**
- Select the language to search
- Type in the exact string. (it is not case sensitive).

Save your changes

4. To save your changes in the existing file, press **Ctrl-S** before exiting.

© 2010 DocuData Software Corporation